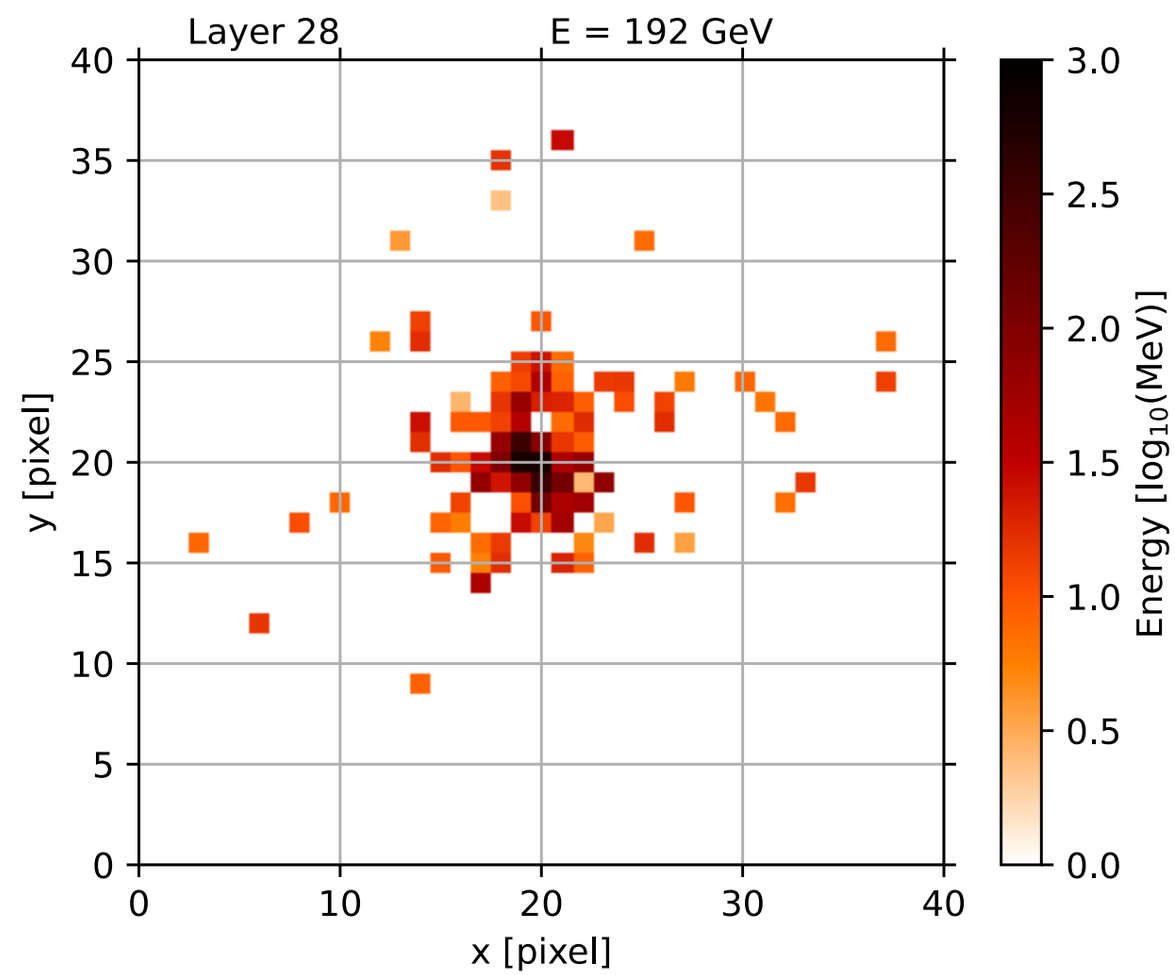
Learning photons

Last time

Learned how to use particle gun to generate, simulate, and reconstruct events (woo)

Prepared particle gun data in ML-friendly format

50k photons
Uniform E in [50, 500] GeV
theta = 20’ (endcap), phi = 0

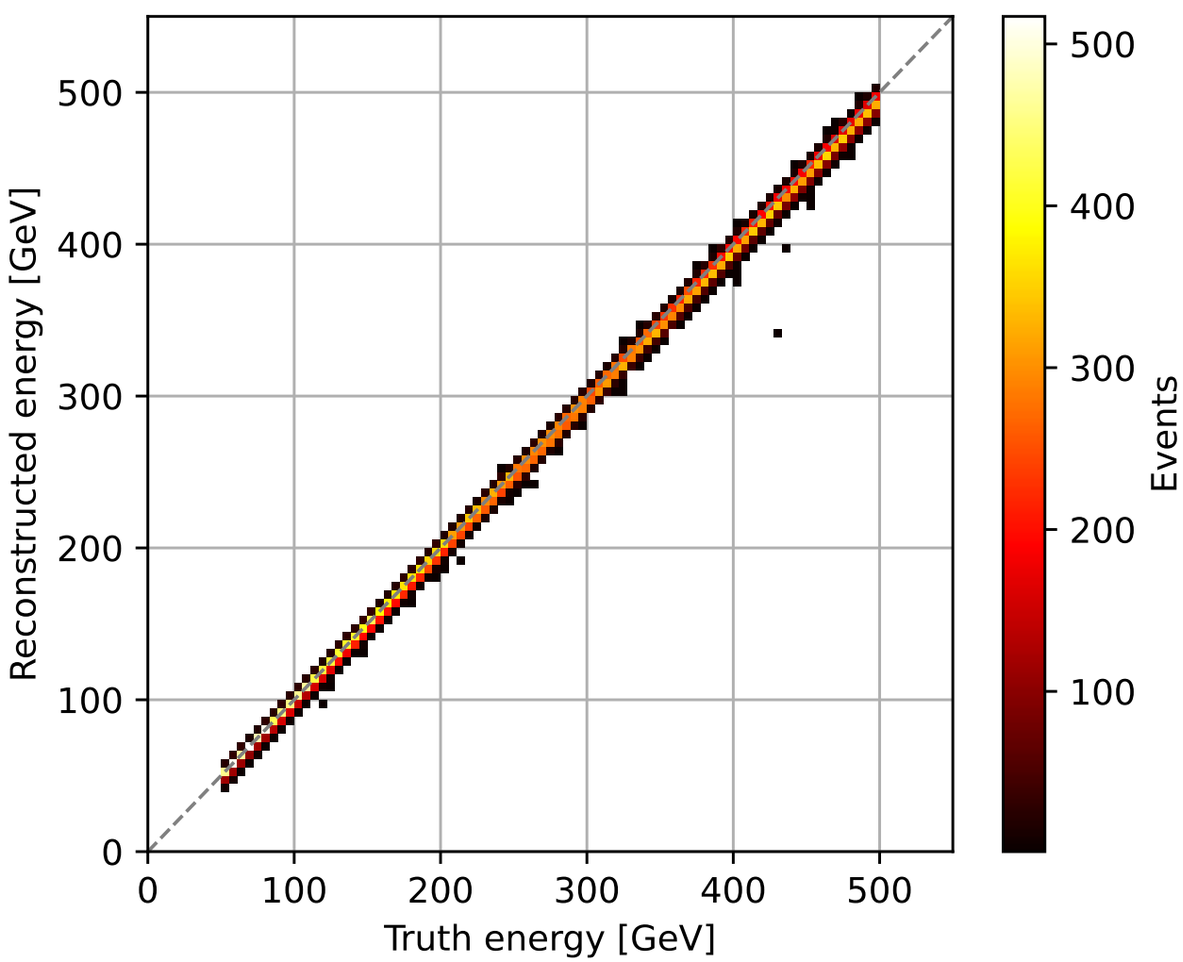


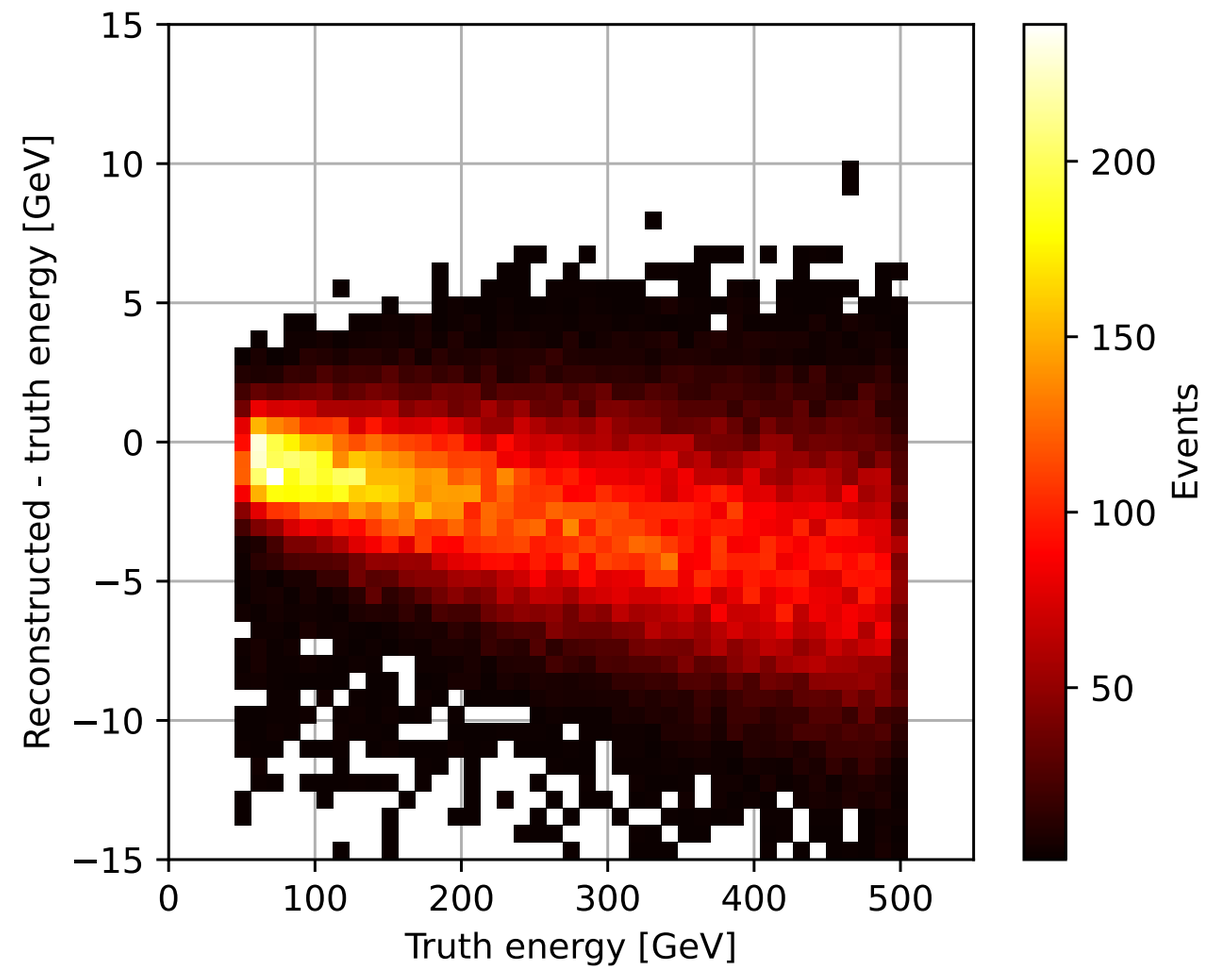
Playing with numpy / pytorch
to regress energy from cells

Essentially no calibration

Trial: sum of calo cells

A baseline: just sum the energy of all ecal cells

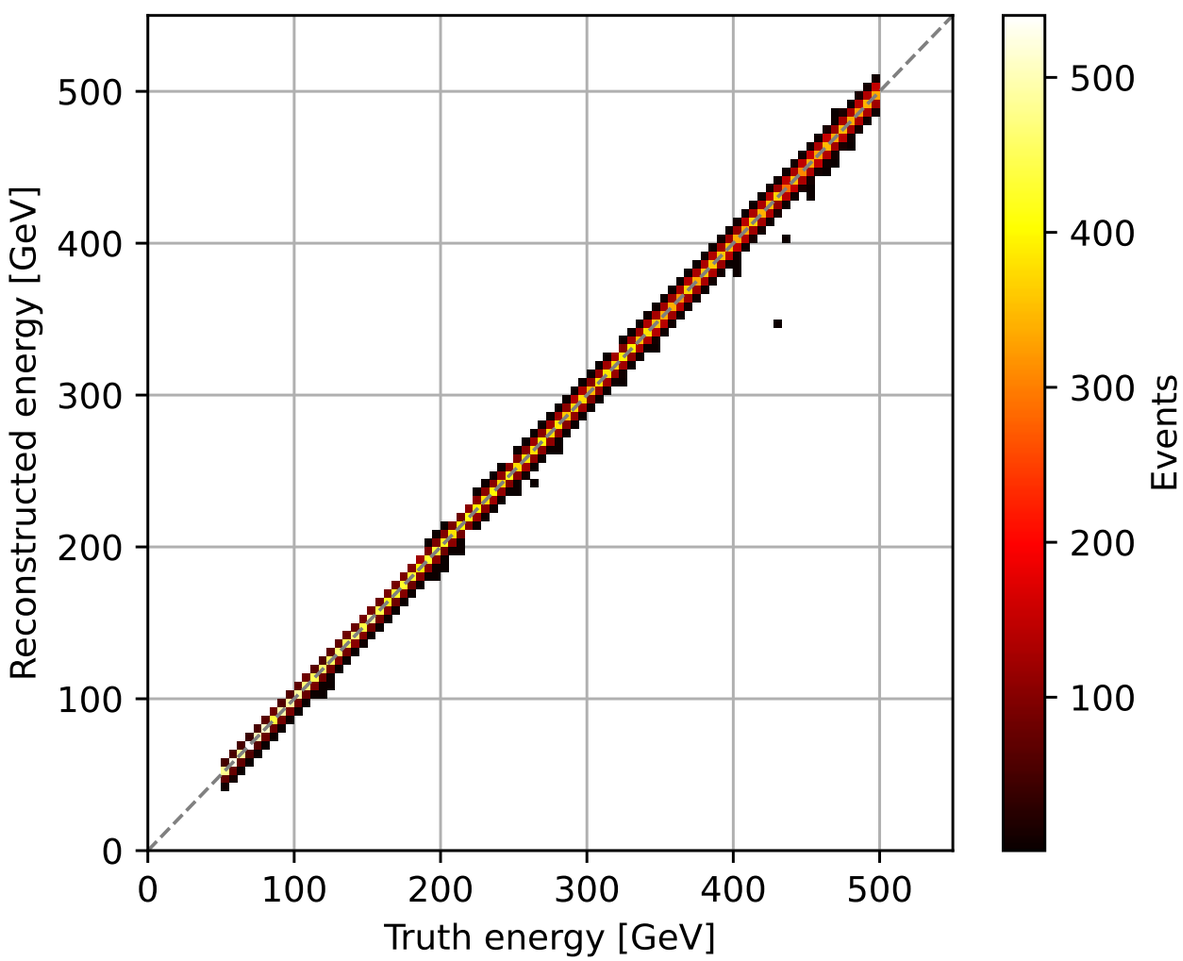


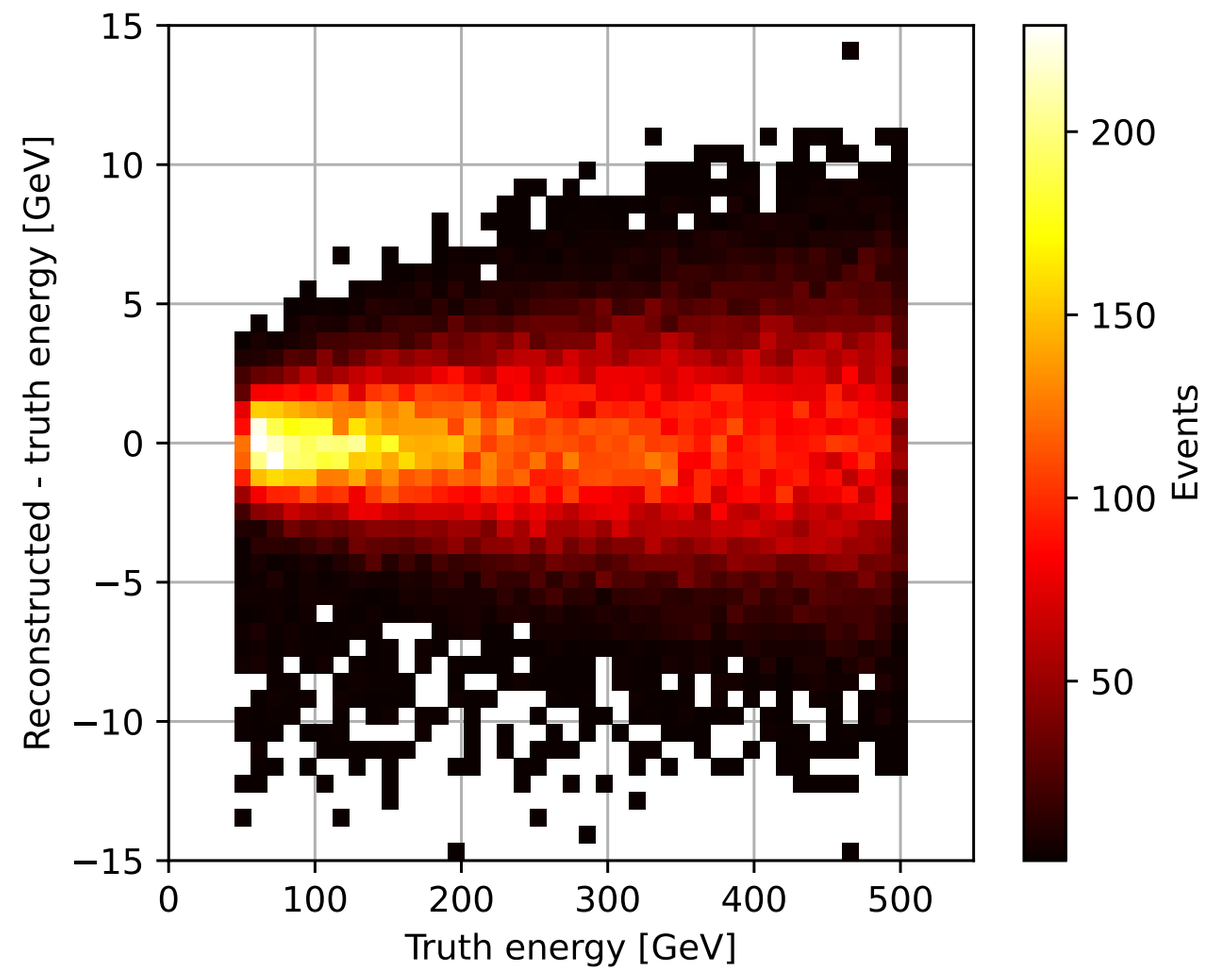


Pretty good! Not perfect

Trial: α*(sum of calo cells) + β

A trial: sum the energy of all ecal cells, with an overall scale/offset



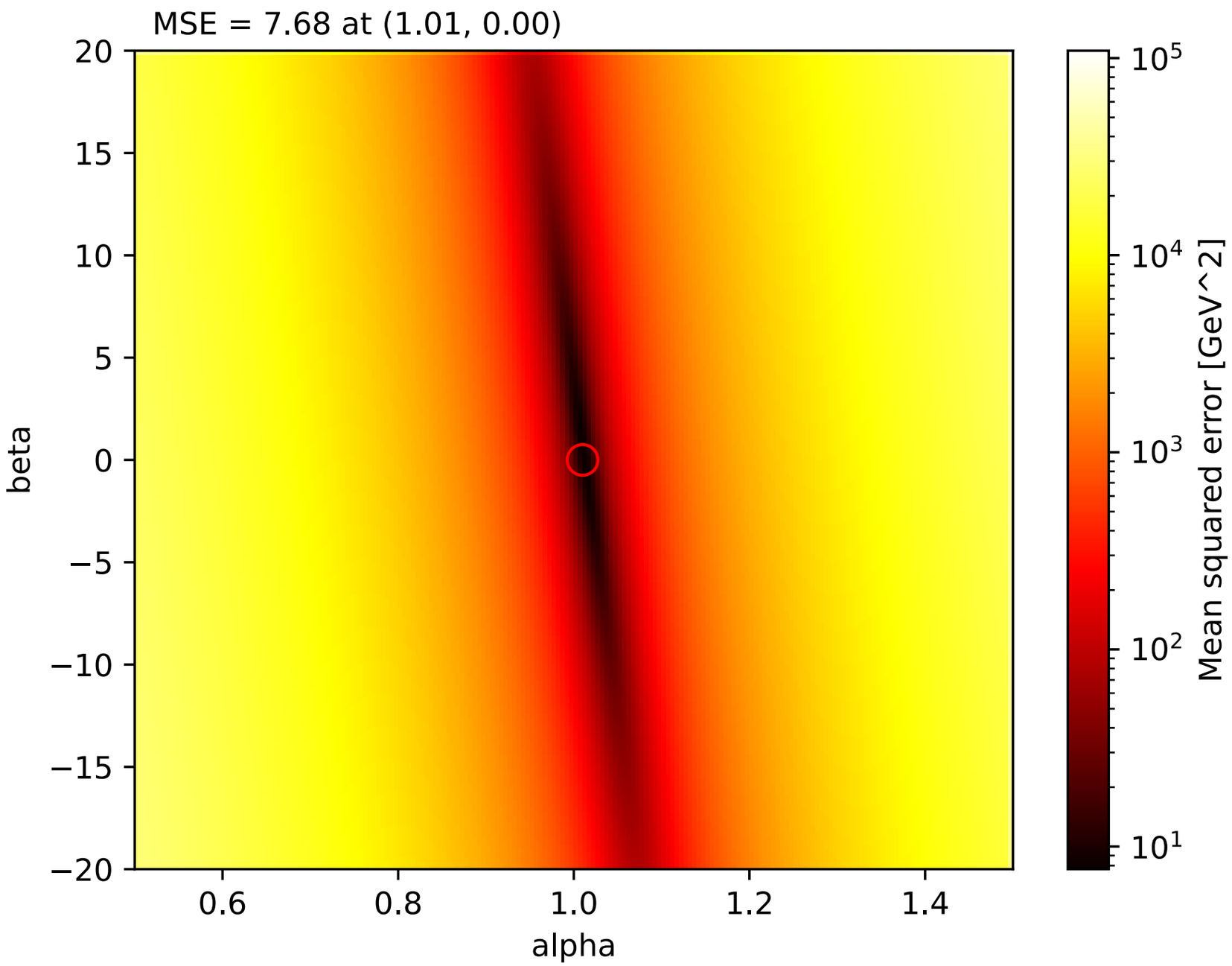


Pretty good! An improvement for sure

Trial: α*(sum of calo cells) + β

Minimum of mean squared error (MSE) at alpha = 1.01, beta = 0

$MSE(\alpha, \beta) = 1/n * \Sigma_i (\alpha*(\text{sum of calo cells}) + \beta - E_{i,\,true})^2$



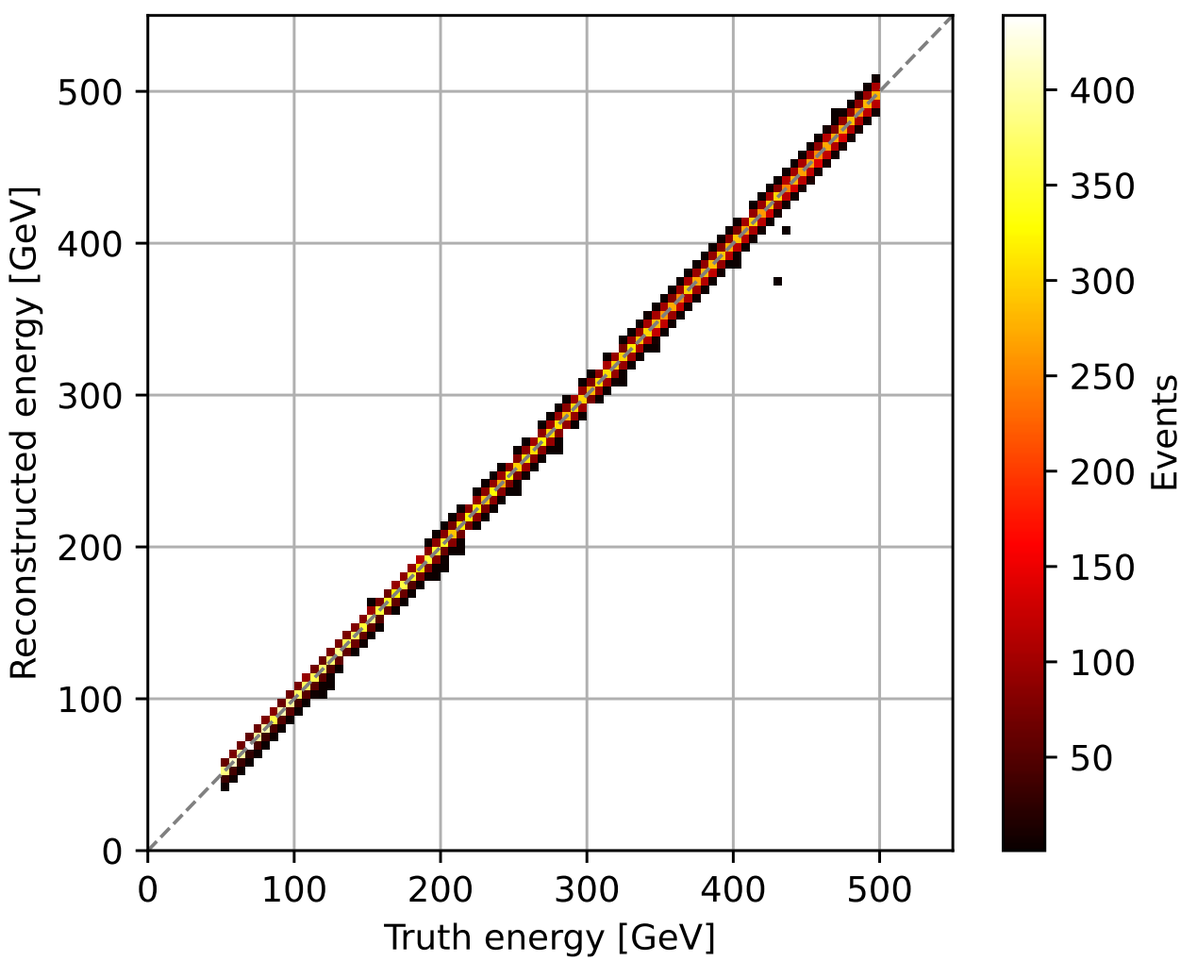
Don't need much code to find minimum in a 2D space

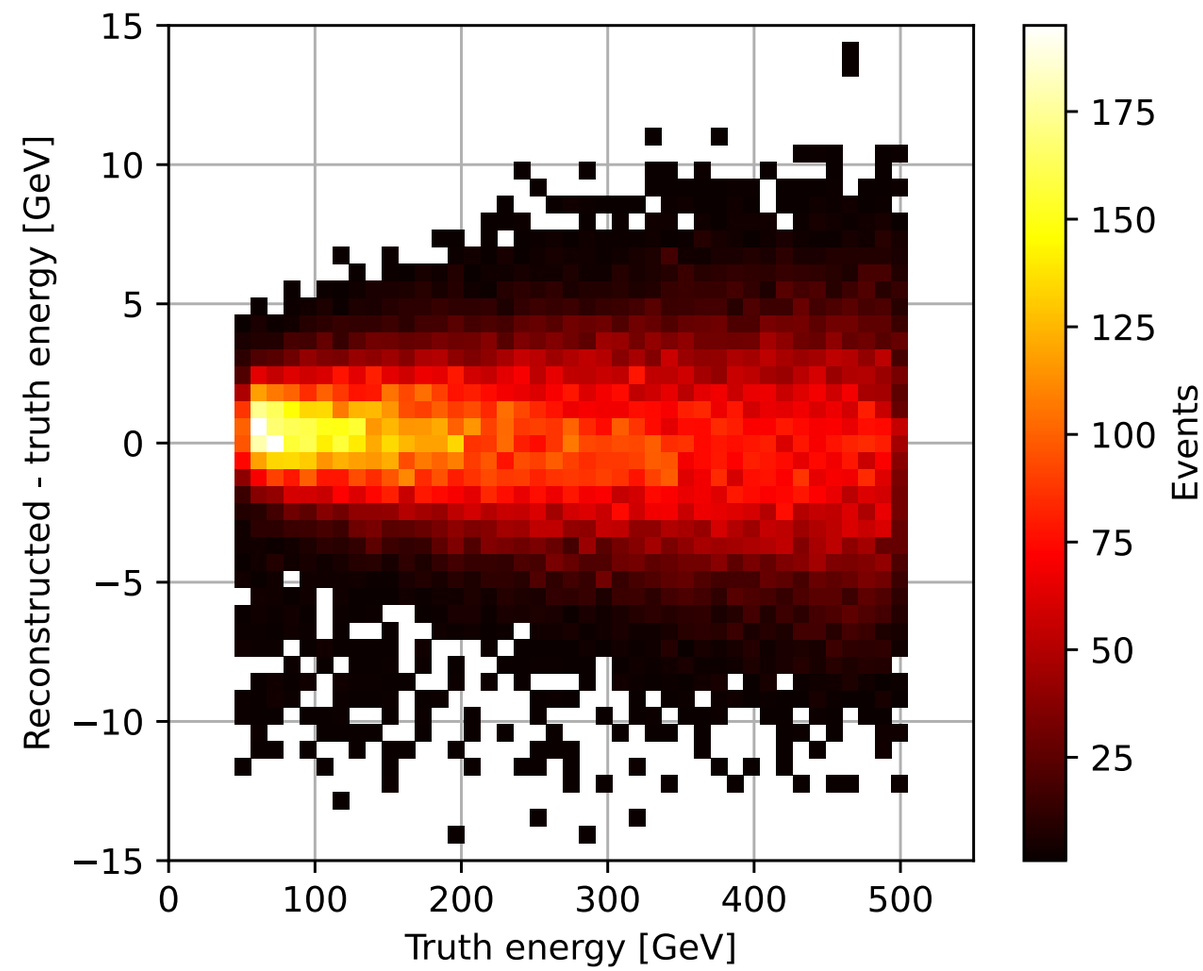
For reference: sqrt(7.68) = 2.77

Mean-squared-error: maybe not the best choice?

Trial: sum of a_i*(calo $layer_i$ cells) + b

A trial: sum the energy of all ecal cells, with a per-layer scale/offset



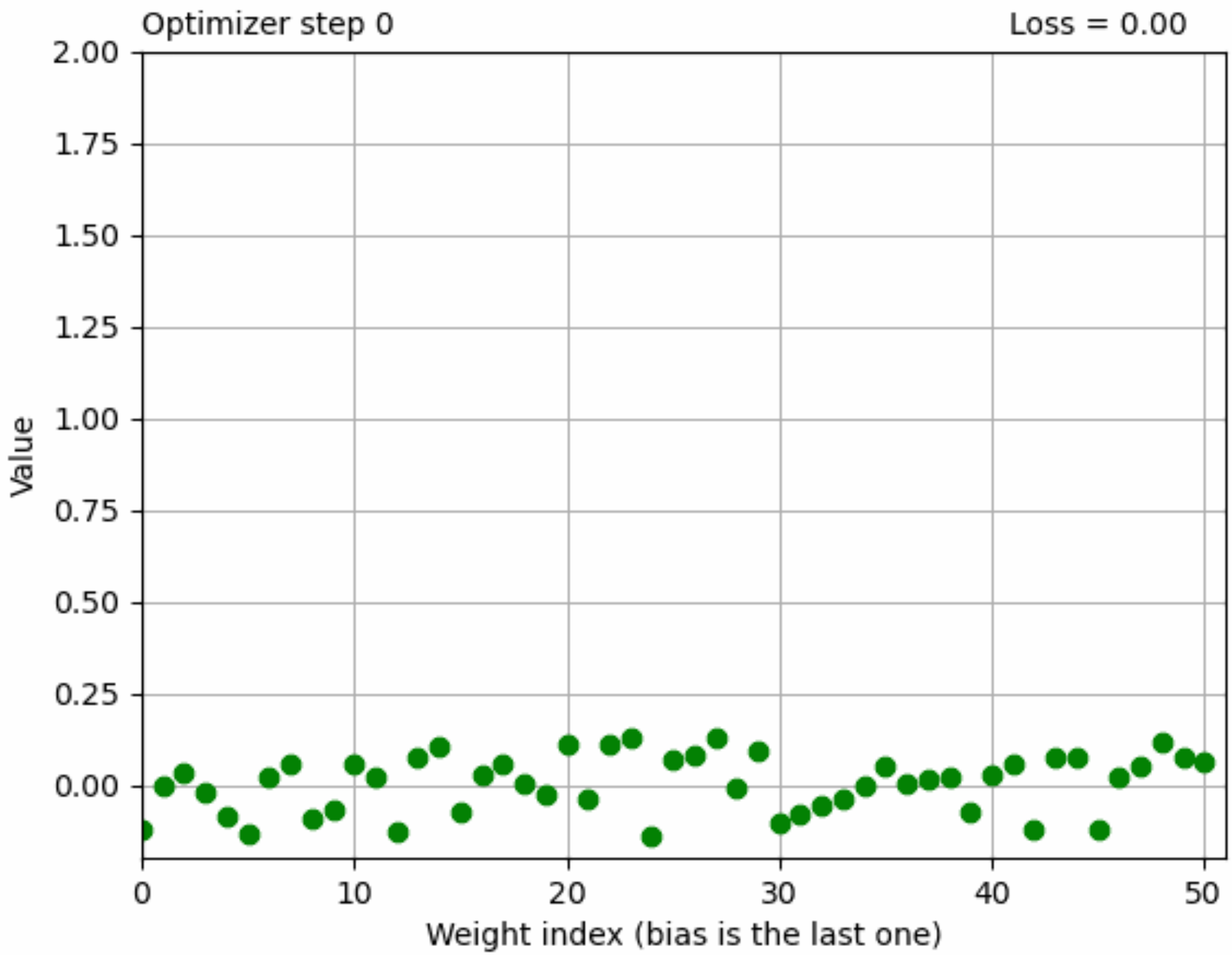


"Neural net" with no hidden layers!
1 weight (calibration) per layer

```
Net:
Sequential(
  (0): Linear(in_features=50, out_features=1, bias=True)
)
N(parameters): 51
```

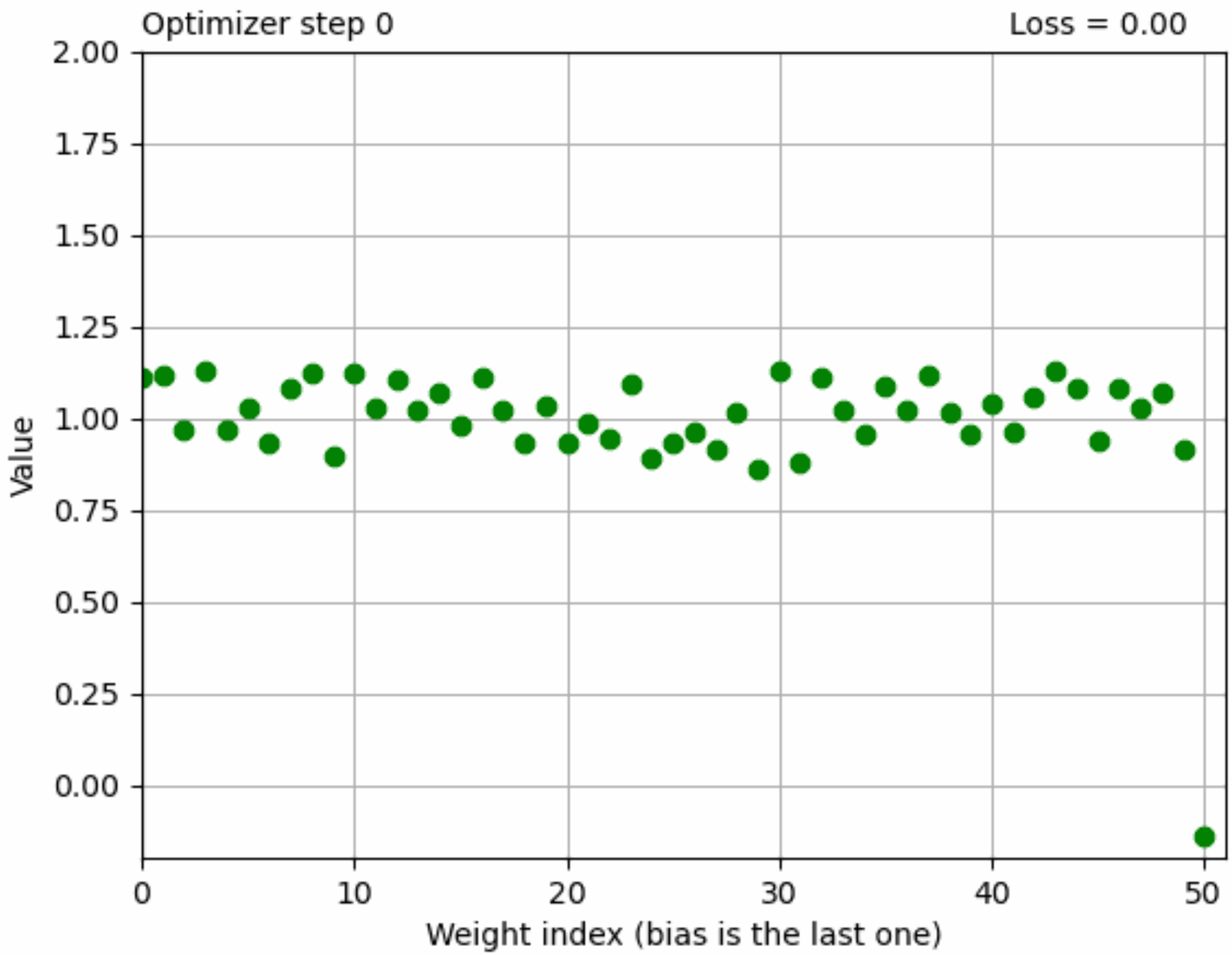
Trial: sum of a_i*(calo $layer_i$ cells) + b

Calibration constants (weights) as a function of time, initialized at 0



Trial: sum of a_i*(calo $layer_i$ cells) + b

Calibration constants (weights) as a function of time, initialized at 1



Summary and plans

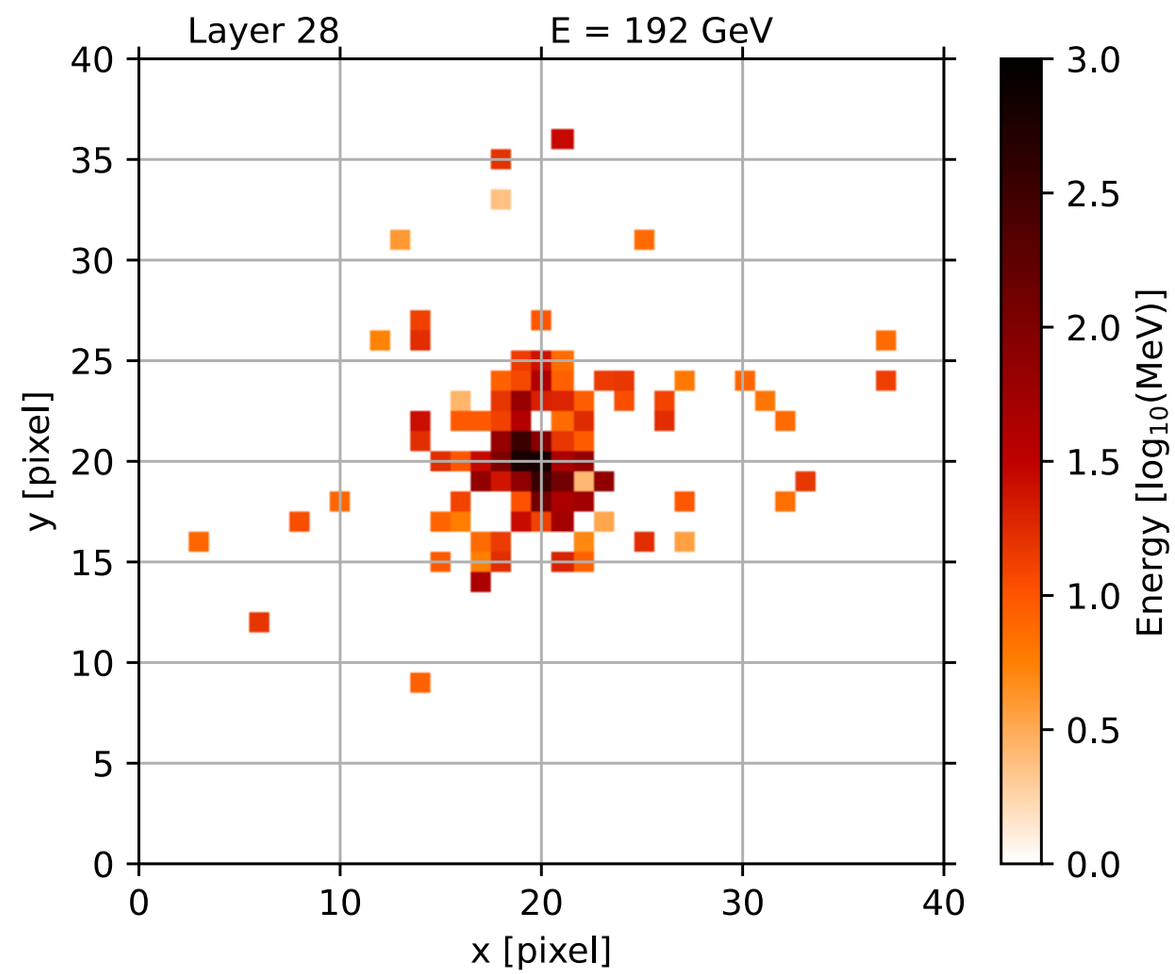
Playing with photon energy regression. Nice improvement with global calibration!

Per-layer calibration doesn't help much. Could try more complex NN of layer info

Or could try using full granularity of the calo: 50 layers * N * N cells

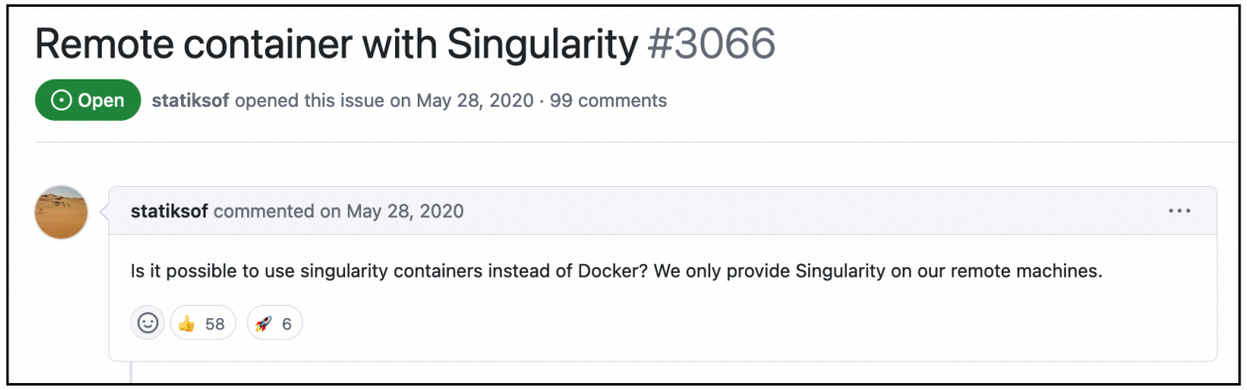
For reference: 50 * 40 * 40 = 80_000

Can also try an easier task if this is too hard lol. e.g. classifying photon vs. pi0, or photon vs. neutron, or …



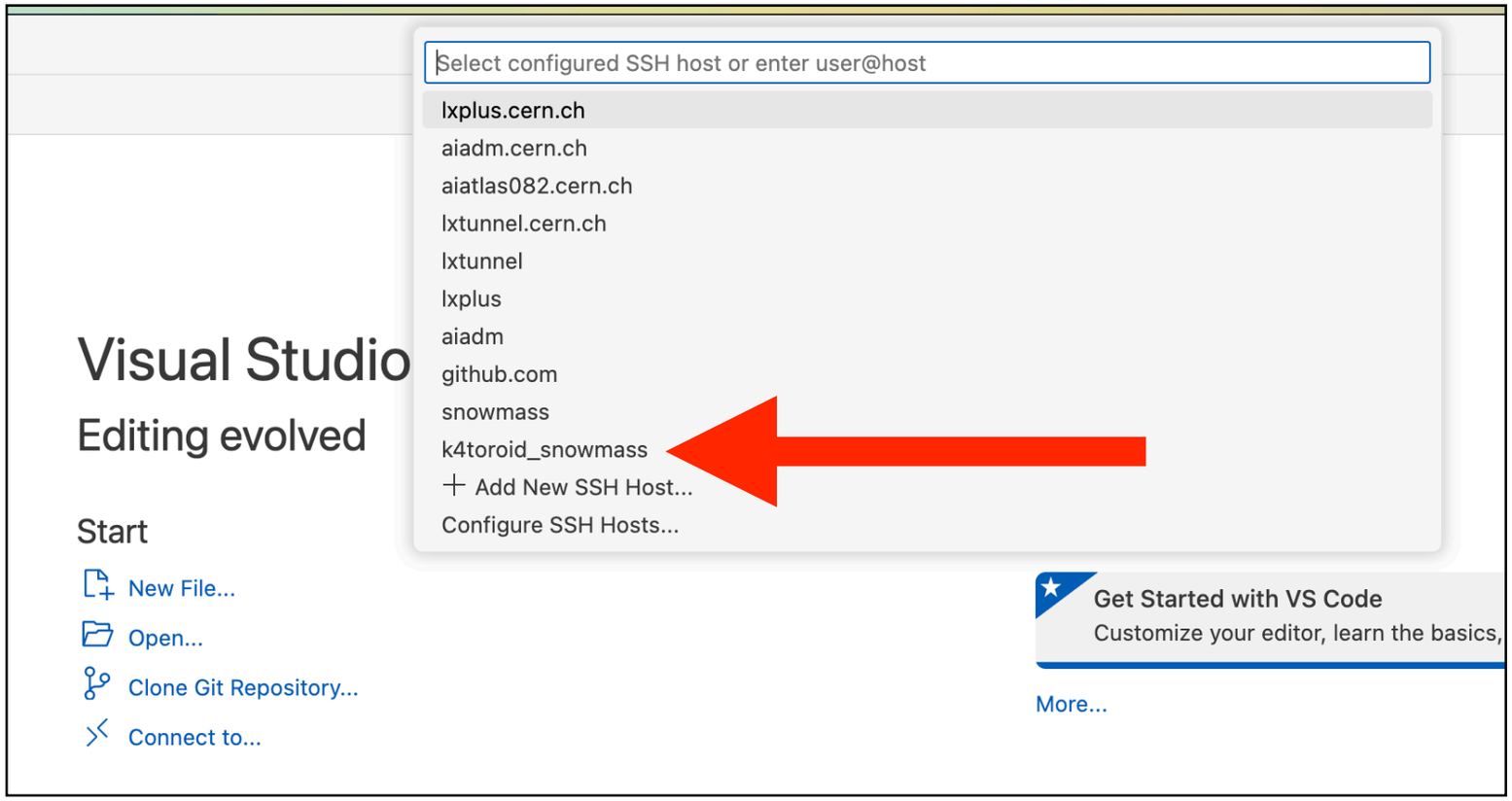
Bonus / past

Bonus: VSCode snowmass/apptainer



Helpful issue: https://github.com/microsoft/vscode-remote-release/issues/3066

```
Alexanders-MacBook-Pro-3:~ alexandertuna$ tail -n 6 ~/.ssh/config
Host k4toroid_*
  RemoteCommand apptainer run -B /tank/data/snowmass21/muonc:/data -B /work/$USER:/code -B /home/$USER -B /work/$USER /home/tuna/k4toroid.sif
  RequestTTY yes
Host snowmass k4toroid_snowmass
  HostName login.snowmass21.io
  User tuna
Alexanders-MacBook-Pro-3:~ alexandertuna$
```



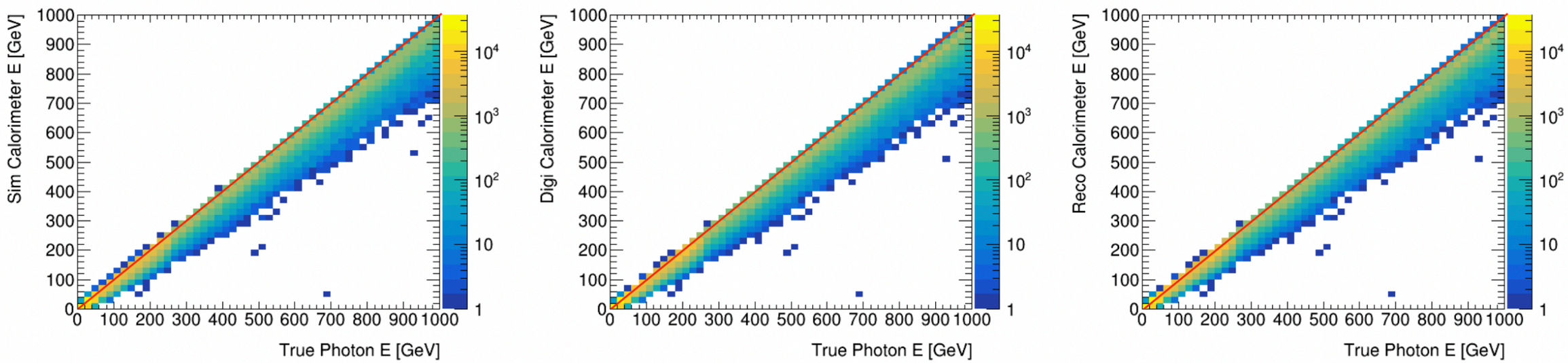
This worked for me on Friday but not Saturday lol

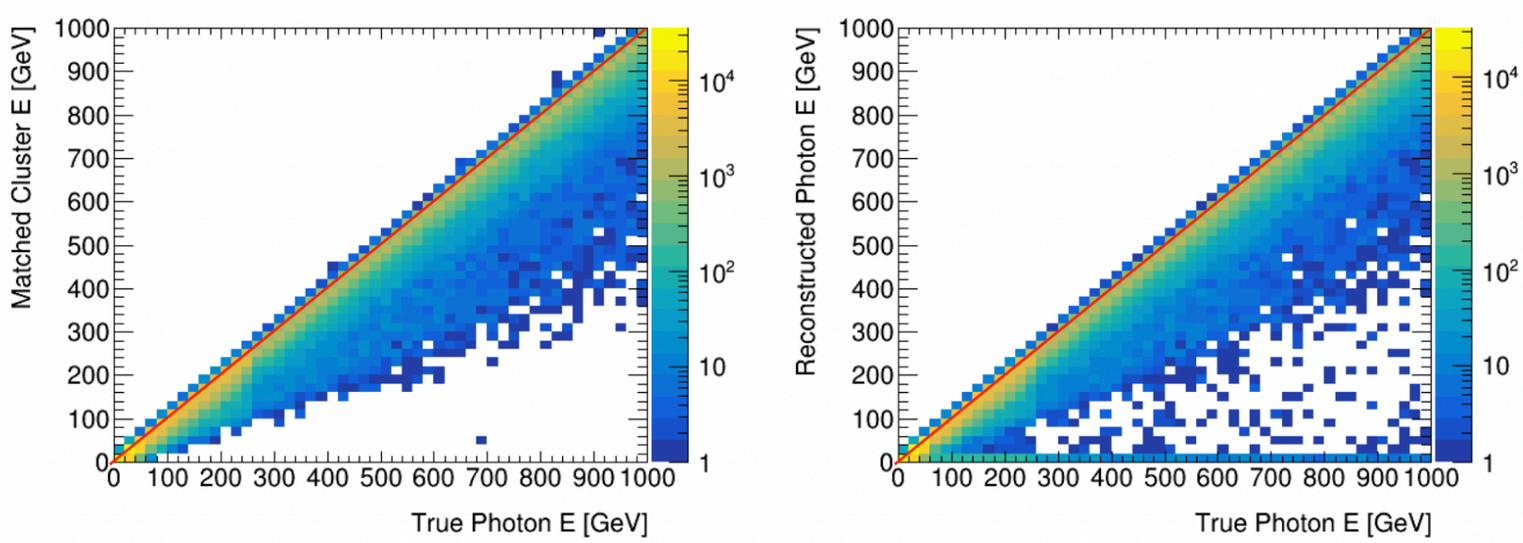
I'm still figuring it out

Tova inspo: indico.desy.de/e/43470/

Photons: ecal

Measured Energies

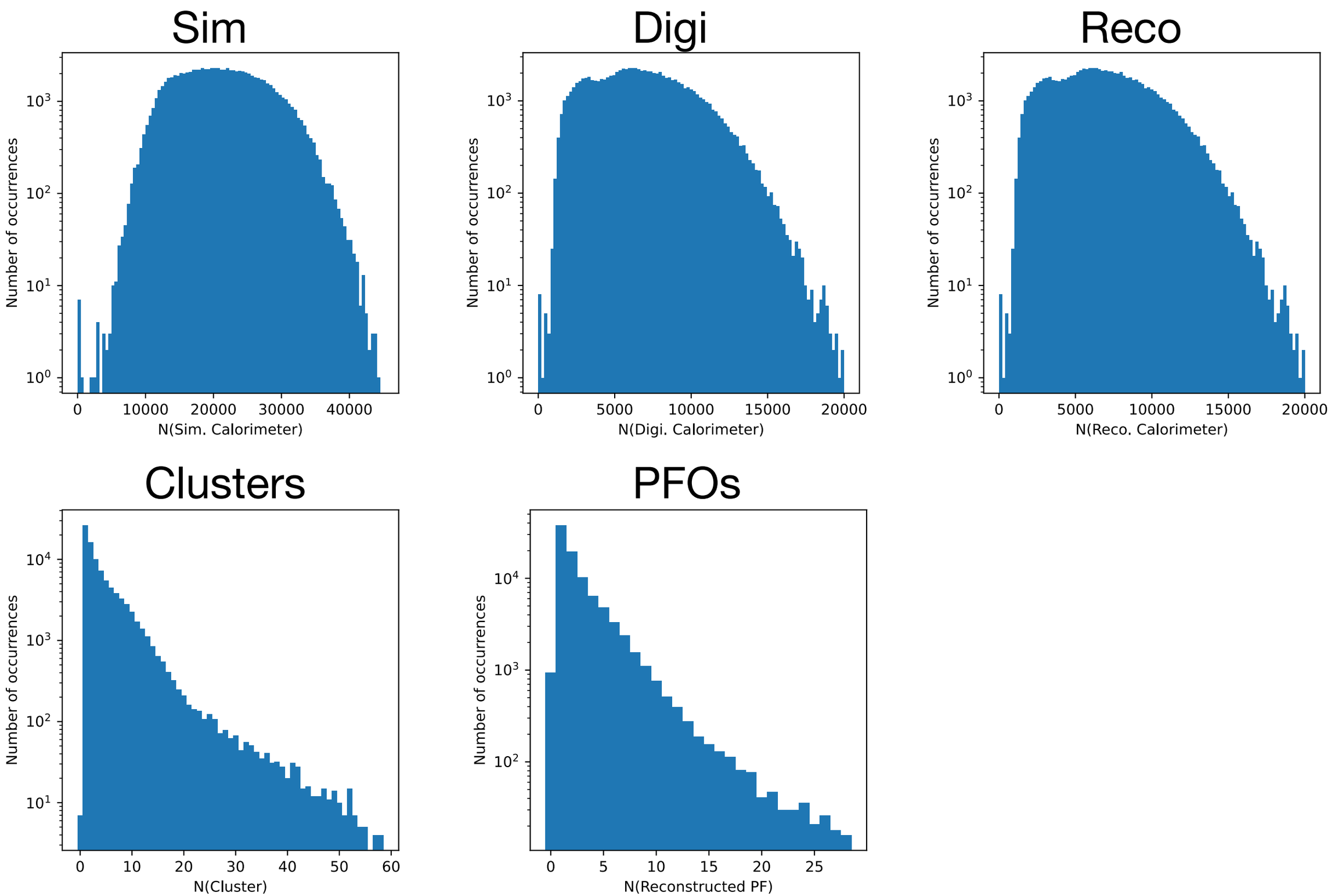




- Interesting that there are some very off-axis ones now even at the Sim level
- Also interesting that photon is not the same as cluster. The key difference is that the reco photon requires the pfo object to have type photon

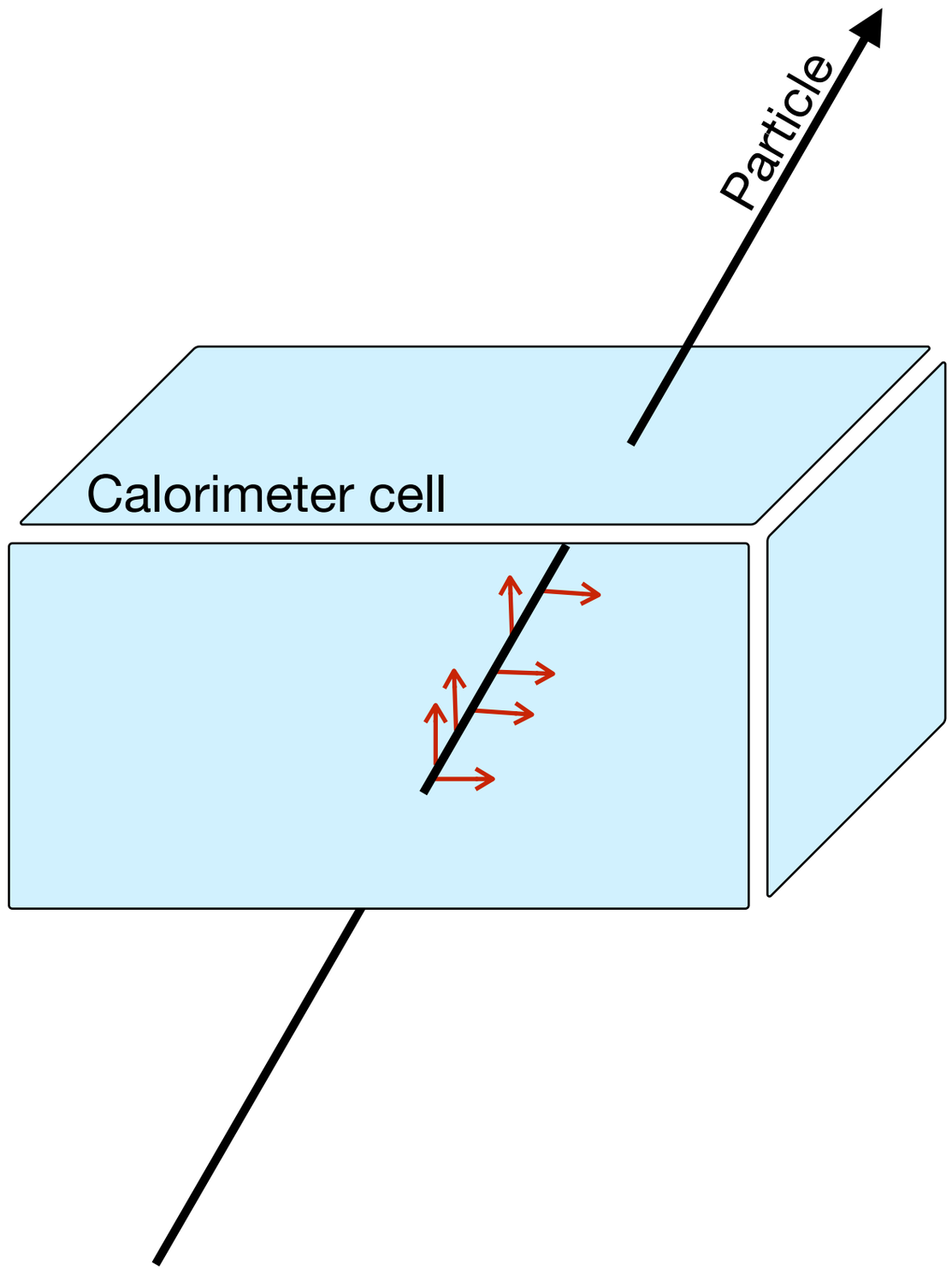
4

Same approach, but for neutrons



Alex's best guess

What's a `SimCalorimeterHit` ?



→ Energy lost by a parent particle within the calorimeter cell, represented by truth (sim.) particles (`MCParticles`)

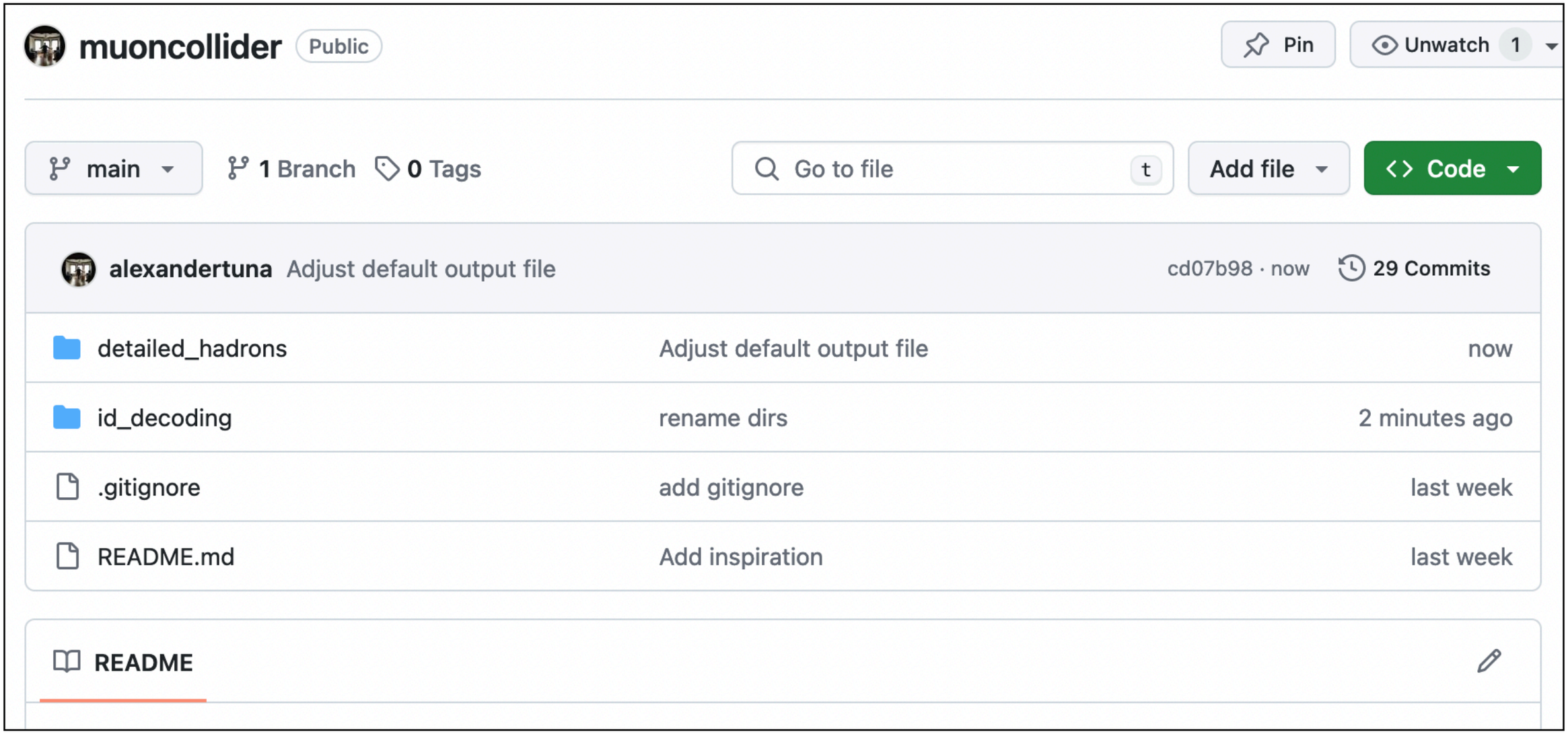
`SimCalorimeterHit` is the list of all `MCParticles` which deposit energy in a given calorimeter cell

`SimCalorimeterHit` is therefore the best energy measurement that cell can possibly achieve

The next stage is digitization, which includes a minimum energy threshold, realistic energy smearing, and other effects

Alex’s GitHub for standalone studies

https://github.com/alexandertuna/muoncollider



Trying to have 1 standalone directory per study / thing

Should I put spaces before and after an arithmetic operator?

How many newlines should precede a function definition?

What should I do if my line is more than 80 characters long?

Should I use single-quotes (') or double-quotes ("))?

Should I use a backslash?

Should I include a trailing comma in my big list of things?

Have you ever …

… cared about coding style?

Use black: github.com/psf/black

And care less!

```
> python -m pip install black
```

Black is the uncompromising Python code formatter. By using it, you agree to cede control over minutiae of hand-formatting. In return, Black gives you speed, determinism, and freedom

```
Apptainer>
Apptainer>
Apptainer> python -m black detailedHadrons/main.py
reformatted detailedHadrons/main.py

All done! ✨ 🍰 ✨
1 file reformatted.
Apptainer>
Apptainer>
```

Use black: github.com/psf/black

Black is a popular choice in Alex's experience

Used by

The following notable open-source projects trust *Black* with enforcing a consistent code style: pytest, tox, Pyramid, Django, Django Channels, Hypothesis, attrs, SQLAlchemy, Poetry, PyPA applications (Warehouse, Bandersnatch, Pipenv, virtualenv), pandas, Pillow, Twisted, LocalStack, every Datadog Agent Integration, Home Assistant, Zulip, Kedro, OpenOA, FLORIS, ORBIT, WOMBAT, and many more.

The following organizations use *Black*: Facebook, Dropbox, KeepTruckin, Lyft, Mozilla, Quora, Duolingo, QuantumBlack, Tesla, Archer Aviation.

Examples of what black does

```
# in:

j = [1,
     2,
     3
]

# out:

j = [1, 2, 3]
```

```
# in:

ImportantClass.important_method(exc, limit, lookup_lines, capture_locals, extra_argument)

# out:

ImportantClass.important_method(
    exc, limit, lookup_lines, capture_locals, extra_argument
)
```

Bonus / future

Using pandas.DataFrame

Enforcing a tabular data structure

Writing to pandas.DataFrame

Slow!

Modularizing

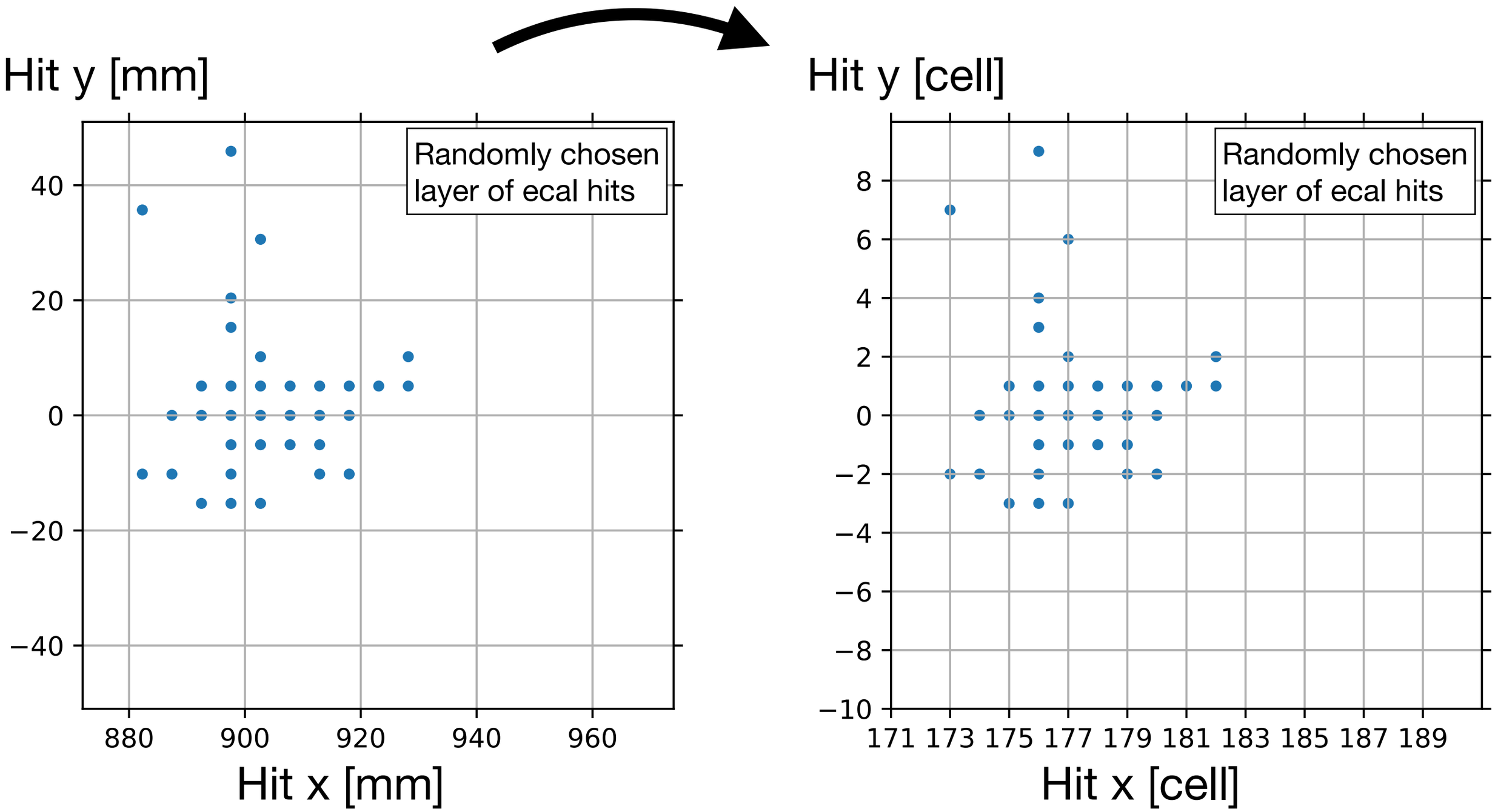
Good for becoming a library
Good for timing measurements
Good for parallel code
Good for readability (hopefully!)

Type-hinting

Testing with pytest

Units: convert from mm to cell number

"Easy": divide by cell size and integerize



Units: convert from cell number to slope?

Divide radial position by z

Not appropriate for image-style architecture!
Cell-spacing of different layers changes in slope units

Probably appropriate for more advanced architecture like graph NN

Units: convert from cell number